REMARKS

The examiner is thanked for the performance of a thorough search. By this amendment, Claims 32, 37, 39, 44, 46, 51, 53 and 58 are amended and no claims are added or cancelled. Hence, Claims 32-62 are pending in the application.

The amendments to the claims as indicated herein do not add any new matter to this application. For example, support for the term "build environment" in the amended claims is found at least in paragraphs [0177]-[0181] of Applicants' specification. Also, the originally-filed independent computer-readable medium claim (i.e., Claim 20, which is now canceled) recites instructions of a build environment that are executed by one or more processors and cause the one or more processors to perform the recited steps.

Furthermore, amendments made to the claims as indicated herein have been made to exclusively improve readability and clarity of the claims and not for the purpose of overcoming alleged prior art.

Each issue raised in the Office Action mailed August 30, 2007 is addressed hereinafter.

I. ISSUES RELATING TO CITED ART

Claims 32-62 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over U.S. Patent Application Publication 2003/0172135 to Bobick et al. ("*Bobick*") in view of U.S. Patent Application Publication 2002/0144248 to Forbes et al. ("*Forbes*"). This rejection is respectfully traversed.

A. CLAIM 32

Amended Claim 32 recites:

> A method of a build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:
> **the build environment compiling source code files into one or more executable file modules;**

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);
the build environment creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;
wherein the software package is created based on at least one of a feature, characteristic, or purpose;
the build environment creating metadata for a first module, of the one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;
the build environment inserting the metadata of the first module into the software package; and
the build environment gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, **by**
(a) receiving a list of dependent modules for a given process or DLL of the first module;
(b) **storing, in the metadata of the first module**, dependency information for the dependent modules in the list, **wherein the dependency information includes API names and versions that the process or DLL depends on**;
(c) **collecting additional dependency information from one or more modules specifications that are separate from the list of dependent modules**, wherein the additional dependency information includes API names and versions that the process or DLL depends on; and
(d) storing the additional dependency information in the metadata of the first module.

MPEP § 2142 states that one of three basic criteria that **must be met** in order to establish a *prima facie* case of obviousness is that "the prior art reference (or references when combined) must teach or suggest all the claim limitations" (emphasis added). *Bobick* and *Forbes* fail, both individually and in combination, to teach or suggest at least the above-bolded elements of Claim 32.

1. The cited art fails to teach or suggest "collecting additional dependency information from one or more modules specifications"

According to Claim 32, there are two different sources of dependency information that a build environment stores: (1) dependency information for dependent modules in the list received

in step (a) of Claim 32; and (2) dependency information in one or more module specifications, as recited in step (c) of Claim 32.

The Office Action concedes that *Bobick* fails to disclose "additional dependency information documented in one or more modules specifications." Specifically, on pages 5-6 the Office Action states:

> Bobick failed to explicitly disclose 'additional dependency information document in one or more modules specifications', in the sense of inter-node and intra-node package version dependencies. (Examiner believes Applicant is referring to dependencies that dependent modules in the package have, i.e., 'nested dependencies.' Applicant's Specification [0095])

The Office Action then cites paragraph [0013], [0045], and [0060] of *Forbes* for disclosing "nested dependencies."

However, paragraph [0095] of the specification describes package dependency information (i.e., dependency information at the level of packages), not the API name and version dependency information as recited in Claim 32. Claim 32 clearly recites additional dependency information that (a) is stored in metadata of a first module and (b) includes API names and versions on which a process or DLL, of the first module, **depends** (see, e.g., paragraphs [0178]-[0181] of the specification). Claim 32 does not refer to package-level dependency information (e.g., inter-node and intra-node package dependency). Therefore, the Office Action rejects Claim 32 based on an erroneous interpretation of Claim 32.

Even if Claim 32 recited nested dependencies, *Forbes* still fails to teach or suggest that the recited additional dependency information is collected from one or more **modules specifications**. Presumably, the Office Action analogizes the manifest file 207 of *Forbes* with the "module specifications" of Claim 32. This is incorrect. The manifest file 207 is clearly **not** a module specification. The cited paragraphs of *Forbes* merely teach that the manifest file 207 "describes the distribution unit to manage the installation, execution, and uninstallation of

software packages on a computer" (paragraph [0013]). A package manager parses a manifest file "to learn if the software package depends on any additional components" (*Id*). More specifically, paragraph [0047] of *Forbes* states:

> The manifest file 207 also contains entries for software that is required to run CoolestApp but which is not included in the distribution unit file 205. Such required software represent "dependencies" and frequently include such items as language libraries and common object class libraries. A dependency can also be another software package.

In order to read on "module specifications", the manifest file 207 of *Forbes* must be a source from which dependency information for a particular module is **collected** by a build environment. That collected dependency information must then be **stored**, by the build environment, in metadata of the particular module, which metadata is inserted into a software package that the build environment creates.

Fundamentally, *Forbes*, like *Bobick*, is unrelated to **creating** software packages and **gathering** API dependency information. Instead, *Forbes* is directed to **installing** and **executing** software packages, which are activities that occur **after** a software package is created. To be consistent, because the manifest file 207 includes dependencies for an application, the manifest file 207 should be equated to the recited metadata of Claim 32. However, *Forbes* fails to teach or suggest (a) how the manifest file 207 is initially generated and (b) the sources from which the dependency information is collected. Therefore, *Forbes* fails to teach or suggest where the dependencies originate, much less that dependencies are collected from one or more module specifications.

2. Bobick *fails to teach or suggest the recited build environment performing the recited steps*

Present Claim 32 recites that a build environment performs the recited steps. One of the recited steps is "compiling source code files into one or more executable file modules." In order

for Claim 32 to be unpatentable, one or more references **must teach or suggest** that the same build environment that compiles source code into executable file modules **also** performs the other recited steps, such as creating a software package, creating metadata for one of the modules, inserting the metadata into the module, and gathering API dependency information. Because neither *Bobick* nor *Forbes* is even concerned with a build and/or development environment, it is not surprising that neither *Bobick* nor *Forbes* teaches or suggests such a build environment.

3. Bobick *fails to teach or suggest "compiling source code into one or more executable file modules"*

The Office Action asserts that *Bobick* discloses "compiling source code into one or more executable file modules" of Claim 32 in paragraphs [0339], [0359], [0360]-[0365] and [0367] of *Bobick*. This is incorrect. In the Response to Arguments section, the Office Action:

> maintains that Bobick discloses [0359], assets 240 are discovered in the source environment…creates an intermediate representation (compiled). The broadest reasonable interpretation of compiling is to change, and in this case source code is changed / grouped into an intermediate representation and packaged for deployment.

The Office Action contends that *Bobick* at least suggests source code is changed. However, the Office Action fails to show where source code is even mentioned.

The Office Action presumably equates the assets 240 of *Bobick* with the recited "source code" of Claim 32. However, in providing examples of assets (i.e., network and/or application components), *Bobick* fails to even mention source code. Additionally, reference to "creates an intermediate representation" in paragraph [0359] cannot be reasonably analogized to compiling source code into executable files because the "intermediate representation" referred to in *Bobick* is "a graph with nodes and edges, each of the digital asset identifiers corresponding to one of the

nodes of the graph, the edges representing the topographical relationship" (paragraph [0203]; emphasis added; see also paragraphs [0224]-[0227]).

Indeed, *Bobick* is not concerned with a build environment that compiles source code files into executable file modules. Instead, *Bobick* is concerned with packaging assets (e.g., applications and program modules, i.e., items that have already been compiled) in one environment in such a way that the assets may also be used in a target environment. In all, none of the cited portions of *Bobick* teach or suggest that source code files are **compiled** into executable file modules, as expressly claimed.

The Office Action additionally cites paragraph [0085] of *Forbes* for teaching "compiling" as recited in Claim 32. However, that paragraph merely states that an OSD formatted manifest file

> provides instructions that can be used to locate and install only the required software components depending on the configuration of the target machine and what software is already present. The OSD formatted manifest file also can be embedded in an archive file, such as a Java Archive (.JAR) file, or a composite, compressed file, such as a cabinet (.CAB) file, that contains the component's distribution unit to form a distribution unit file.

This paragraph mentions nothing about compiling. "Compiling," in any reasonable computer science meaning, refers to transforming human-readable computer program statements into code that a machine can directly execute or interpret.

JAR is a file format based on the popular ZIP file format and is used for aggregating many files into one. Although JAR can be used as a general archiving tool, JAR was developed so that Java applets could be downloaded to a browser in a single HTTP transaction, rather than opening a new connection for each piece. A simple reference to JAR files does not teach or suggest "a build environment compiling source code files into one or more executable file modules" as recited in Claim 32. JAR is not an intermediate program file format and is not executable code.

Like *Bobick*, *Forbes* is not concerned with a build environment that ***creates*** software packages. Instead, *Forbes* is directed to installing and executing software packages.

4. Bobick *fails to teach or suggest storing version information, in the metadata of a module, of APIs on which a process or DLL of the module depends*

Claim 32 recites "storing, **in the metadata of the first module**, dependency information for the dependent modules in the list, wherein the dependency information includes API names and **versions that the process or DLL depends on**" (emphasis added). In the Response to Arguments section, the Office Action cites various portions of *Bobick* (i.e., paragraphs [0223], [0461], [0463], [0487]) that refer to version information. However, such version information is of an asset (i.e., the alleged first module) that is to be transferred from a source environment to a target environment. Such version information is **not** stored in the metadata of a module, much less stored by a build environment that also compiles source code files into one or more executable files, as recited in Claim 32.

Furthermore, even if *Bobick* discloses that version information is stored in the extended environment 220 (i.e., the alleged metadata) of a digital asset (i.e., the alleged first module) as contended in the Office Action, *Bobick* **fails** to teach or suggest that such version information is of other digital assets (i.e., the alleged APIs) upon which the digital asset (i.e., the alleged "process or DLL") **depends**, as Claim 32 would require. *Forbes* also fails to teach or suggest this feature of Claim 32.

Based on the foregoing, *Bobick* and *Forbes* fail to teach or suggest, both individually and in combination, all features of Claim 32. Accordingly, Claim 32 is patentable over *Bobick* for at least the reasons provided above. Reconsideration and withdrawal of the rejection of Claim 32 under 35 U.S.C. § 103(a) is therefore respectfully requested.

B. CLAIMS 37, 44, 51, 58

Claims 37, 44, 51, and 58 are independent claims that include some of the same features discussed above with respect to Claim 32. For example, Claims 37, 44, 51, and 58 recite that (a) a build environment performs the recited steps, (b) source code is compiled into one or more executable files, and (c) version information of APIs upon which a module depends are stored in metadata of the module. Therefore, Claims 37, 44, 51, and 58 are allowable for some of the same reasons that Claim 32 is allowable. Reconsideration and withdrawal of the rejection of Claims 37, 44, 51, and 58 under 35 U.S.C. § 103(a) is therefore respectfully submitted.

C. DEPENDENT CLAIMS

Each of Claims 33-36, 38-43, 45-50, 52-57, and 59-62 is dependent upon one of the independent claims discussed above. By dependency, each of Claims 33-36, 38-43, 45-50, 52-57, and 59-62 includes some of the same features discussed above with respect to the independent claim upon which it depends. Therefore, each of Claims 33-36, 38-43, 45-50, 52-57, and 59-62 is allowable for the same reasons discussed above for the claim upon which it depends. Reconsideration and withdrawal of the 35 U.S.C. § 103(a) rejection with respect to Claims 33-36, 38-43, 45-50, 52-57, and 59-62 is therefore respectfully submitted.

II. CONCLUSIONS & MISCELLANEOUS

For the reasons set forth above, all of the pending claims are now in condition for allowance. The Examiner is respectfully requested to contact the undersigned by telephone relating to any issue that would advance examination of the present application.

If any applicable fee is missing or insufficient, throughout the pendency of this application, the Commissioner is hereby authorized to any applicable fees and to credit any overpayments to our Deposit Account No. 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Dated: November 28, 2007

/DanielDLedesma#57181/
Daniel D. Ledesma
Reg. No. 57,181

2055 Gateway Place Suite 550
San Jose, California 95110-1083
Telephone No.: (408) 414-1229
Facsimile No.: (408) 414-1076